Applying Integrated Safety Analysis Techniques (Software FMEA and FTA)

November 30, 1998
Quality Assurance Office

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

Applying Integrated Safety Analysis Techniques (Software FMEA and FTA)

Prepared by:

_____ _____

Robyn R. Lutz
Task Lead

Hui-Yin Shaw
Task Co-Lead

Reviewed and Approved by:

_____ _____

Burton C. Sigal
Software Assurance Supervisor

John Kelly
ATPO Software Applications Program, PEM

November 30, 1998
Quality Assurance Office

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

Applying Integrated Safety Analysis Techniques (Software FMEA and FTA)

TABLE OF CONTENTS

Summary

This report describes recommended process features for integrating the software and system safety techniques of SFMEA/FMEA and SFTA/FTA, with examples drawn from applications to the New Millenium Program (NMP) Mars Microprobe Project (MM) and the Earth Observing System's Microwave Limb Sounder (MLS). The process integrates Software Failure Modes and Effects Analysis (SFMEA) and Software Fault Tree Analysis (SFTA) into the system-level hazard analysis. The main lessons learned from the applications are discussed. These include (1) flexible use of the techniques, (2) a risk-driven rather than sequential approach, (3) "zoom-in/zoom-out" use of SFMEA/SFTA, (4) SFMEA and SFTA as complementary techniques, (5) preserving traceability, and (6) applicability to fault protection software. Since one obstacle to expanded use of SFMEA/SFTA is the lack of web-based support, an experimental web-database tool with improved data-sharing and data-search capabilities for SFMEAs was developed and demonstrated. This work was funded by NASA Code Q RTOP UPN 323-08-5I.

1. Introduction

This report describes the integration of the existing software safety techniques of Software Failure Modes and Effects Analysis (SFMEA) and Software Fault Tree Analysis(SFTA) into the system engineering process and demonstrates their use on one spacecraft project and one instrument project. The lessons learned from these applications are discussed. These include (1) flexible use of the techniques, (2) a risk-driven rather than sequential approach, (3) "zoom-in/zoom-out" use of SFMEA/SFTA, (4) SFMEA and SFTA as complementary techniques, (5) preserving traceability, and (6) applicability to fault protection software. A web-database tool was demonstrated which improved data-sharing and data-search of the SFMEA information, thus enhancing the software and system safety analyses.

This report is a product of a research effort funded by the NASA Software Independent Verification and Validation Facility as a Center Software Initiative. The work reported here is an extension of previously reported work on integrating software and system safety. Following Leveson [8], software safety is defined to be freedom from undesired and unplanned events that result in a specified level of loss. In the Deep Space 1 Project Safety Plan [1], safety-critical software is defined as software that can command a hazardous function to happen or prevent a hazard from occurring.

Current needs for higher reliability, reusable software, rapid development, and innovative software architectures have focused attention on improving our requirements and design analysis techniques. In earlier work for this software initiative, we investigated the use of the SCR* (Software Cost Reduction) requirements toolset to support software safety by creating requirements models and performing a design logic check against requirements and rules on two components of the New Millenium Program's Deep Space 1 spacecraft software [10]. The research reported here shifts attention to the design phase. It describes ways to integrate software and system safety analyses of the software design, with application to the Mars Microprobe (MM) software and the Microwave Limb Sounder (MLS) instrument.

SFMEA is a design analysis method that explores the effects of possible software failure modes on the system. SFMEA is an extension of the hardware FMEA, which has been a standard engineering activity since the 1970's. SFMEA has been used on flight projects at JPL (Galileo, Cassini, NMP MM), primarily to verify the correct functioning of system-level fault protection software.

Briefly, SFMEA is a structured, table-based process of discovering and documenting the ways in which a software component can fail and the consequences of these failures. It is most frequently used during the design phase, but was also used during the requirements phase of Cassini. The SFMEA process is guided by a set of standardized failure modes (e.g., "Wrong timing of data," "Abnormal process termination") which the analyst considers in turn.

The SFMEA is a form of forward (bottom-up) analysis in that the process traces the propagation of anomalies from causes (failure modes) to local (subsystem or component) effects to global (system) effects. One of the key benefits of SFMEA has been its usefulness in discovering unknown failure

modes by means of this structured analysis. See [11] for a more complete definition of the SFMEA process.

Fault Tree Analysis (FTA) is a hazard analysis technique that works backward (top-down) from an identified undesired event or hazard to discover its possible causes [8, 15]. It is widely used in the systems and hardware areas [2, 6, 7], and has been successfully applied to software as well [9].

The effectiveness of the bottom-up SFMEA has been found to be increased by combining it with a top-down SFTA. The SFMEA/SFTA combination allows the underlying combination of circumstances that enable the failure mode to occur, as well as the likelihood of the identified failure mode to be evaluated. The effectiveness of the SFMEA is also increased by integrating it with existing system FMEA or system FTA. NASA's Software System Safety course recommends that a SFMEA and SFTA be performed and spends a day-and-a-half of the four-day syllabus on those safety techniques.

The work described below on the MM Project yielded process recommendations for how the software and the system safety analysis can be more effectively integrated.

2. Process Description

2.1 System-Level Hazards Analysis

On Mars Microprobe, the project had already produced a system-level fault-coverage table, which we used as the hazards analysis baseline. The table was included in the Mars Microprobe Spacecraft Design document [14]. The table, entitled "Fault Coverage," listed for each key function (e.g., telecommunications) the types of faults that could occur (e.g., loss of uplink, loss of downlink, etc.) and the coverage that is provided for each of these fault types. Some of the fault types involved hardware failure, some involved software failure, and some involved both. The coverage for some faults was, at least in part, software-based. Response to these faults usually entailed software control of hardware devices. The coverage for other faults was hardware-based, e.g., a watchdog timer.

2.2 Software FMEA

In this step, we verified the adequacy of the software handling of the fault types described in the system-level fault coverage table. We did this by performing a SFMEA for three of the critical functions: Telecommunications, impact-related activities (Impact Loss Monitor, Accelerometer-Off Response, and Impact Detection and Penetration Measurement event), and Water Detection.

The Telecommunication subsystem (Telecom) is a crucial component in communicating science and engineering data from the microprobe to the orbiting spacecraft (the orbiter) for transmission back to the Earth. The Telecom consists of an antenna and a microprocessor controlled transmitter/receiver located on the aftbody of the microprobe. It gathers temperature and pressure data from various locations on the aftbody, combines this data with data supplied by the Advanced Microcontroller (AMC) subsystem, and formats and encodes the data for transmission to the orbiter.

Impact Detection is critical because the Mars Microprobe is a single stage from separation from the orbiter to its impact with the Martian surface. It has no active control, attitude, or propulsive systems. The Impact Detection and Penetration Measurement portion of the Entry, Descent and Impact sequence will initiate and take impact and penetration measurements, transfer these science and engineering data to the Telecom and set a task completion mark to signal for the start of Landed Mission which includes soil and water experiments.

The Water Detection Experiment is critical because the data from the water experiment has been identified as "Critical Science Data" that the mission is required to return. Shortly after impact, a small drill will collect a subsurface soil sample and return the soil to the water experiment's sample cup. The water experiment is designed to detect whether subsurface ice is present and to measure the temperature at which any water vapor is released. Onboard fault protection software exists to aid in recovery from failures during this phase of the mission. The key findings from the SFMEA and the recommended design changes are summarized in Section 3.1.

The SFMEAs were reviewed by several Mars Microprobe engineers. Their feedback, as well as changes to the design, were then incorporated into a final SFMEA, again delivered to the project. The Mars Microprobe project performed system-level interface FMEAs. We looked at the interface FMEAs, but since they were hardware-based analyses, there was little overlap in issues with the SFMEAs. The Project had also initially planned to perform additional system-level FMEAs. When these were later descoped, the results of the SFMEAs provided some assurance to the Project regarding the adequacy of the fault protection coverage.

2.3. Integrated Component-Level Safety Analysis

The Telecom entry in the fault-coverage table described above was expanded into a subsystem (also called component-level) FMEA. The Telecommunications subsystem FMEA was reconstructed with information from the Spacecraft Design Document. The subsystem FMEA traced forward from the occurrence of subsystem faults to check whether their effects were acceptable.

We also expanded the entry for Water Detection in the fault-coverage table into a component-level FTA. The FTA traced backward from the occurrence of the faults to check whether their root causes were prevented or mitigated. The main results from these safety analyses are described in Section 3.

2.4. Review

Review of the results was ongoing, as described above. The final version provided to the Project incorporated the comments from the reviews, updated the SFMEAs to match design changes, and corrected errors in the preliminary analysis results. The final version included a description of the issues for additional design analysis and/or testing. Follow-up was recommended to ensure that the final code handles the hazards identified.

3. Applications

This section describes the applications of the integrated software safety analysis techniques to the New Millenium Program Mars Microprobe Project (MM) and the Earth Observing System's Microwave Limb Sounder.

The Mars Microprobe Project, also known as the Deep Space 2 (DS-2) mission, consists of two identical microprobes that will penetrate the Martian surface [13]. The mission will validate technologies which will enable future planetary network missions (e.g. simultaneous deployment of multiple landers, penetrators, etc.) while at the same time collect meaningful science data on Martian soil conductivity, meteorology, and subsurface ice.

Each microprobe consists of two major components: the surface (aftbody) module and a subsurface (forebody) module. The aftbody contains the battery system, telecommunication system (or the Telecom), power turn-on switch, atmospheric pressure sensor system, solar detector and descent accelerometers. The forebody contains the flex deployment system, sample/H20 experiment, power electronic system, soil thermal conductivity experiment, impact accelerometer, instrument electronic system, and AMC. AMC is the Advanced Microcontroller which contains critical software that runs the forebody functions and the related fault monitor and response modules.

The Earth Observing System (EOS) Microwave Limb Sounder (MLS) instrument, currently under development, will support an investigation that will improve understanding and assessment of stratospheric ozone depletion and chemistry, tropospheric ozone distribution and chemistry, and climate change and variability [3]. The MLS instrument will measure naturally occurring microwave thermal emission from the limb of Earth's atmosphere to remotely sense vertical profiles of selected atmospheric gases, geopotential height, temperature and pressure. The MLS instrument will fly on the EOS Chemistry platform to be launched in December 2002. The instrument comprises physical and electronic elements that acquire scientific measurements. It is the successor to the highly successful MLS on the Upper Atmosphere Research Satellite.

3.1. Software FMEA

3.1.1 MM Impact-Related Software FMEA

The FMEA technique was employed in the MM impact-related software safety analysis. The purpose of this exercise was to verify the adequacy of the software handling of impact-related faults described in the system-level fault coverage table. The objectives included identifying critical software failures and assessing the appropriateness of the fault avoidance and mitigation and of the recovery sequence. The impact-related fault monitor and response and the event sequence were evaluated: Impact Loss Monitor, Accelerometer-Off Response, and Impact Detection and Penetration Measurement event.

This study yielded the following results: 1. Clarification of sequence and fault response and clarification of software variable in an anomalous event, 2. Recommendation of test cases, and 3. Identification of a software fault avoidance.

Some key questions involving unclear definition and missing information for anomalous scenarios surfaced during this study. For example, in the case of no impact detection, the transfer of science and engineering data from the forebody to Telecom was not included in the event sequence or in the fault response module. The definition of impact time was unclear in the design document; it wasn't clear what the value of impact time would be in the event when no impact is detected. This impact time is used in the Soil Thermal Conductivity Experiment sequence. These findings were considered by the Design Engineer for inclusion in a future document update.

Failure entries in the SFMEA marked with medium and high criticality were evaluated for the appropriateness of fault identification, avoidance and/or mitigation. Several medium to high criticality failures were identified as verifiable in test (i.e., tests can be performed to verify the absence of potential faults). Recommendation was made to the Project to include these test cases.

In this study, most of the highly critical failures involve software hang-up. Depending on when in the event sequence the hang-up occurs, such hang-up may cause missed science experiment data (with fault protection and recovery sequence in working order). A discussion with the software developer, a subject matter expert, was conducted to identify sources leading to possible software hang-up. It turned out that a potential software hang-up exists when a particular assembly command is executed at the same time that a certain built-in interrupt timer goes off (therefore the software hang-up may be sporadic and difficult to debug). A fault avoidance method was recommended by the subject expert to disable (and later enable) the timer before execution of this particular assembly command.

3.1.2 MM Water Experiment Software FMEA

Eight issues were raised in the preliminary SFMEA (see Table 1). All were resolved, most by subsequent re-design. (Note that the re-design was initiated by the Project independently of the analysis done here.)

Table 1. Water Experiment SFMEA Issues

SFMEA Issue:	Resolution:
Some data not used?	All data used; documentation updated
Data dictionary needed	Table of Mission Sequence Variables added
Add two watchdog timers or timeouts?	Functionality handled by current electronics
Possible unintended design constraint	Re-design eliminated software at issue
Flag not used or needed in current design	Re-design creates and uses set of markers (checkpoints)
Duration of sample heating ambiguous	Values updated
Algorithm for detection of sample not stated	Clarified in update
Add nominal rate at which data buffer fills	Re-design adds limit on time available to fill data buffer

3.2 Component-Level Safety Analysis

3.2.1 Integrating System and Software FMEAs

A subsystem (component-level) FMEA table was constructed for the Telecom subsystem with the information obtained from the system design document [14]. This exercise showed that performing FMEA on a critical (sub)system, component, or function could help identify areas requiring fault monitor and response modules. These monitor and response modules are identified in the Failure Detection/Correction column of the FMEA table. These modules can involve hardware, software or both. Table 2 shows an excerpt from the Telecom subsystem FMEA.

Table 2. Excerpt from Telecom Subsystem FMEA

Function/ Event	Failure Mode	Failure Detection/ Correction	Effect	Criticality	Remark
Telecom Sequence	General recoverable Telecom software or hardware failure	(M) Telecom Watchdog Timer [h/w]; (R) Telecom Hardware Reset [s/w]; (R) Telecom Safing [s/w]	The Telecom mission sequence is resumed at the next appropriate mark point.	High	Protects against Telecom software or hardware failure.
Forebody/ Aftbody Communi-cation	Loss of Forebody	(M) Umbilical Loss [h/w & s/w] (R) Telecom Autonomous Control [s/w] M: fault monitor R: fault response	Telecom will enter autonomous mode: Go to listen mode and sample and store aftbody science and engineering data once an hour. AMC can command Telecom out of this mode at any time - by AMC resetting Telecom autonomous watchdog timer	High	Protects against loss of mission due to failure in AMC to communicate with Telecom (caused by forebody under-voltage, tether breakage, or AMC loss)

The subsystem FMEA table and the subsequent fault monitor and response design lay the groundwork for the top-level (requirements/design-level) software safety analyses. In our study, three of the software-controlled monitor and response modules from the Telecom subsystem FMEA were expanded into the top-level software FMEA.

The three modules studied were the Umbilical Failure monitor, the Telecom Autonomous Control response for the Loss of Forebody failure, and the Telecom Hardware Reset response for general

recoverable Telecom software or hardware failure. These modules were chosen because the loss of Forebody communication with Telecom could have a major impact on the primary science objective of this mission, and the Telecom Hardware Reset response module plays an essential role in the general recovery of Telecom failures.

At the time when we were conducting this SFMEA study, the MM Project was transitioning into test phase. As it turned out, all the highly critical software failures identified in this study were verifiable in test for software implementation correctness. Therefore, recommendation was provided to the project for inclusion of these failure scenarios in their test cases.

Some of the lessons learned during the application of the integrated safety techniques on Mars Microprobe were considered to be transferable technology. These lessons are summarized in Section 4 and formed part of the input to the process for MLS analysis. Since MLS is at an earlier phase of development than Mars Microprobe, the system-level analysis performed to date has been primarily hardware-oriented. Techniques such as SFMEA that focus on the software's contribution to system fault coverage can be used to expand analysis of critical components or capabilities.

3.2.2 Integrating Component-Level FMEAs and FTAs with Top-Level Software FTA

In the EOS MLS project, the component-level FMEAs were reviewed for potential failures where software might play a part. The components studied included the Giga Hz (GHz) Module which contains the Antenna Actuator Assembly and the Switching Mirror Assembly, and the Tera Hz (THz) Module which contains the Scan/Switching Assembly [4]. It was found that similar failure types appeared among the component-level FMEAs, e.g., "Loss of Bus Synchronization" failure mode appears in GHz and THz FMEAs. When appropriate, we generalized these common failures when performing the top-level SFTA. Each of the selected failures became the top-level hazardous event (root node) of a SFTA. For each root node hazard, we worked backwards (top-down), expanding each sub-node until a basic fault event was reached (a leaf of the fault tree), or until no further analysis could be performed [16].

A discussion with the software engineer on the SFTAs proved beneficial. Several follow-up items and a few further analyses were proposed as a result of this meeting. Feedback from the software engineer was incorporated into the final SFTA. The software engineer felt that this was a worthwhile exercise in evaluating all the possible failures/faults and their avoidance and mitigation.

Four MLS component-level FTAs were also reviewed. They include the Antenna Launch Latch, the Antenna Actuator Assembly, and two Scan/Switching Mechanisms. Faults (or leaf nodes of Fault Trees) that may be attributed to software failure were identified. These selected component faults became the root node hazard (or the root of a fault tree) for the next lower-level FTA, in this case, the top-level software FTA. The same software FTA procedure was performed as in the component-level FMEA to top-level software FTA study.

It is worthwhile to note that in system and component level FMEAs and FTAs, the analyses are most frequently hardware-parts oriented. Software and operational attributed faults at the system or

component level are often not being considered. For example, “commanding failure” (software and operational) can be included as a sub-node to the node "Latch Fails to Open” in the Latch FTA.

This work was performed based on the latest versions of the MLS component-level FMECAs and FTAs available at the time of this study (mostly draft/working versions). The final component-level FMECAs were later reported in the EOS MLS System-Level FMECA. The draft component-level FTAs were internal working document and memos. The resulting top-level software FTAs have identified the following types of fault tree leaf nodes:

- Software faults verifiable in test: e.g., command format error, telemetry transmission scheme compatibility
- Lower-level (source code) analysis required: e.g., the need to determine software/mechanism behavior resulting from out-of-range command parameters and for consideration of adding exception handling or software assertion for fault avoidance or mitigation
- Operational errors: e.g., incorrect command sequence, wrong or out-of-range command parameters
- Hardware-attributed faults: e.g., electronic noise induced command bit drop in the bus
- External faults: e.g., spacecraft telemetry pickup error. Note, the spacecraft and the MLS instrument are separate entities in this problem domain.

The MLS top-level software FTAs also validated the adequacy of software commands for the control of hardware mechanisms. Furthermore, one mission critical hardware component was identified (the real-time interrupt generator). While this hardware component has a redundant unit, the instrument recovery mechanism requires verification. Additional findings from the top-level software FTA include:

- Further analysis required: e.g., instrument/software behavior resulting from executing a command in an inappropriate mode; instrument/software behavior resulting from execution of corrupted command; communication error recovery scheme
- Follow-up cases were identified: e.g., to determine the appropriate reset mechanism for Remote Interface Unit of the MLS instrument
- Workarounds identified: e.g., re-send command when command is corrupted; the flexibility of software allows for in-flight software change to respond to faulty register memory map.

3.3 Web-Based Application

Data from safety analysis can become massive and unmanageable in a relatively short period of time. To manually sift through these data for specific information is tedious and carries the risk of inadvertently overlooking critical data.

A web-based database application can link upper-level safety analyses to lower-level analyses. For example, a system FMEA can be linked to a component-level FMEA, and these can be linked to specific software and hardware FMEAs. This allows better traceability of safety-critical elements from the system to the software and the hardware, and to their fault avoidance or mitigation

strategies. Depending on the project's needs, links to safety requirements, tests, event sequences or design can also be implemented. The tool can also provide options to do a search in an area of interest, such as failure criticality, failure modes, or affected requirements. Restricted editing capabilities can support on-line updates. When developing an application such as this web-based tool, the tool developer should work with the intended application users to ensure that the tool meets the needs of the intended users.

In our experimental tool development, we were limited to the tools that were available to us. Cold Fusion from Allaire was used to create dynamic Web pages and to interact with Microsoft Access database. The performance of this experimental tool was not a consideration. This web-based safety analysis application stores the FMEA results and provides the ability to do search and report on the FMEA entries. Each FMEA can be stored as a table in the web database application. For example, a subsystem FMEA for Telecom is stored as one table, and a software FMEA is stored as another table. A user can search on criticality, testability, and /or failure modes in any selected combination of tables. An additional search option that would be useful is "Affected Requirements." However, this feature was not implemented in the experimental tool because the information was not included in our FMEA study.

The experimental web-based tool demonstrated that we could quickly locate the information we needed to help us do our work. For example, we could quickly identify failures that were identified as testable in the earlier safety analyses and use this information to follow up in test planning and test verification when appropriate. A web-based safety analysis information retrieval tool can benefit projects in several ways. Most importantly, the safety analysis results can be readily accessible to project developers and analysts for follow-up and further analysis of critical issues. Similarly, safety-critical information can be easily available for anomaly impact analyses and for requirements or design change impact assessments.

4. Lessons Learned

The lessons learned in the application of the integrated software and system safety techniques are summarized below. These are recommended elements of any similar process.

1. Flexible use.

We found that a key advantage of the integrated approach is that the focus of the analysis can be tailored to the needs, phase, and available documentation of the specific project. On Mars Microprobe this meant using the existing hazards analysis work that had been done as a baseline, extending the analysis (via SFMEA) in the directions that were of most concern to the project, and performing component-level general analysis (FMEA/FTA) subsequent to the component-level software analysis (rather than having the broader analysis precede the software analysis, the more common sequencing).

2. Risk-driven

An advantage of the integrated SFMEA/SFTA approach is that it allows a progressive re-focusing of attention on those components or faults that are currently of greatest concern. For one component, a SFMEA followed by a verbal walkthrough with experts (rather than a FTA) resolved the open issues. For another component, a SFMEA was later supplemented by a FTA to follow up on some issues of concern. SFMEA and/or SFTA can be performed only for those components perceived as possibly presenting unacceptable risk, or SFMEA/SFTA can be applied selectively to differing levels of detail on different components, all depending on the project's needs.

3. "Zoom-in/zoom-out" use of SFMEA/SFTA

A consequence of the flexible use of SFMEA/SFTA is that it can provide a "zoom-in/zoom-out" approach to analysis of critical components. Selective targeting of issues of concern, designs that have changed, or areas that raise unresolved questions is possible. A "zoom-in" can be chosen to examine more closely a particular piece of the system or the effect of a particular scenario. Similarly, a "zoom-out" can be chosen to examine a wider piece of the system when the interest is in the component's interfaces rather than in, e.g., a fine-grained description of all possible events leading to a particular state. This capability of SFMEA/SFTA to allow the analysis to be tuned to the evolution of the system is useful. On Mars Microprobe, for example, questions that arose during the initial analysis of one component (Impact Detection) led both to a quick "zoom-out" review of the system-level interface FMEA (to check if there were any related software issues) and to a "zoom-in" on the SFMEA issue regarding software hang-up to identify software fault avoidance.

4. SFMEA and SFTA as complementary techniques

SFMEA and SFTA have a well-established and well-deserved reputation as complementary techniques [5, 12]. Integrating a forward search for the consequences of failure modes (SFMEA) with a backward search for contributing causes has been shown to be successful in identifying inconsistent, missing, and incorrect requirements [11]. In particular, the combination has been used to identify unexpected dependencies and interactions in the system. Some researchers have performed SFMEA as a preparatory activity to fault tree construction (e.g., [12]). Others have recommended first performing a backward search for causes (FTA), and then considering the effects of each failure (as in a FMEA). HAZOP, a safety analysis technique originating in the chemical industry that performs backward analysis first, has been a major influence on efforts to integrate SFMEA and SFTA [8].

On Mars Microprobe we primarily performed SFMEA, since the Project had identified the critical faults that needed coverage in the system. Backward analysis from the faults to their contributing causes (SFTA) was performed less, since the causes of the faults were nearly all hardware or environmental (e.g., landing) failures. These hardware failures had been extensively researched. Since our concern was mainly with the software/system interaction, we were led to forward analyses from system failures to their software (i.e., fault protection) responses.

5. Preserving traceability

Leveson makes a useful distinction between forward and backward searches which trace forward (to effects, e.g., SFMEA) or backwards (to causes, e.g., SFTA) in time, and top-down and bottom-up searches, which involve levels of abstraction [8]. The traceability between the forward and backward search is temporal; the traceability between the top-down and bottom-up search involves refinement. The traceability between SFMEA/SFTA, as discussed above, is usually the temporal movement from failure causes to failure effects.

SFMEA/SFTA can also be performed on a single system at varying degrees of detail, in which case the traceability is between higher-level and lower-level analyses. Because SFMEA/SFTA can be done at several levels of detail, traceability among levels is sometimes explicit. For example, failure effects in a lower level FMEA may be the failure modes in the left hand column of a higher-level FMEA. Similarly, for FTA, a single node in a high-level tree may be broken down into a more detailed FTA.

As Leveson points out, a bottom-up search that examines only the effects of individual component failures on the system may miss hazardous system behavior resulting from combinations of subsystem behaviors. On the other hand, a top-down search may be an inefficient way to determine the effect of a particular component behavior.

6. Applicability to fault protection software

SFMEA/SFTA is a labor-intensive effort, best suited to innovative, poorly understood, or critical components. In this case, our interest was in SFMEA/SFTA as safety analysis techniques, so we chose critical fault protection software for the application. SFMEA is well-suited to analysis of fault protection monitors and responses. For monitoring software, SFMEA was used to check that false-positives were not produced and that adequate reasonableness checks were performed on the values used to make control decisions. For fault protection software that responds to faults, SFMEA was used to check that the effect of the response (e.g., reconfiguration, try-again, etc.) matched the intent of the fault response. The SFMEA results provided some assurance that the fault coverage asserted in the design document was adequate and robust.

Acknowledgments

We thank Sarah Gavit, Kari Lewis, Parviz Danesh, Bob Detweiler, and Robert Nowicki on MM; Gary Lau, Dennis Flower, Marc Walch, Mike Girard, Mark Boyles, Philip Szeto, John Klohoker, and Johnathan Carson on MLS; and Paul Davis for assistance with the web page application.

References

1. Deep Space 1 Project Safety Plan, JPL D-13533, October, 1996.

2. DeLemos, R., A. Saeed, and T. Anderson, "Analyzing Safety Requirements for Process-Control Systems," IEEE Software, Vol. 12, No. 3, May, 1995, pp. 42-52.

3. Earth Observing System, Microwave Limb Sounder Project homepage, http://mls.jpl.nasa.gov/.

4. Earth Observing System Microwave Limb Sounder, System-Level Failure Modes, Effects, and Criticality Analysis, JPL D-16088, Version 1.0, September 1, 1998.

5. Fenelon, P. and J. A. McDermid, "An Integrated Tool Set for Software Safety Analysis," Journal of Systems and Software, Vol. 21, July, 1993, pp. 279-290.

6. Hansen, K. M., A. P. Ravn, and V. Stavridou, "From Safety Analysis to Software Requirements," IEEE Transactions on Software Engineering, Vol. 24, No. 7 July 1998, pp. 573--584.

7. Knight, J. and L. G. Nakano, "Software Test Techniques for System Fault-Tree Analysis," 1997.

8. Leveson, N. G., Safeware: System Safety and Computers. Addison-Wesley, 1995.

9. Leveson, N. and P. R. Harvey, "Analyzing software safety," IEEE Transactions on Software Engineering, SE-9(5), Sept, 1983, pp. 569-579.

10. Lutz, R. and H.-Y. Shaw, "Applying the SCR* Requirements Toolset to DS-1 Fault Protection," JPL D-15198, December 1997, http://eis.jpl.nasa.gov/quality/qadc/software.htm

11. Lutz, R. and R. M. Woodhouse, ``Requirements Analysis Using Forward and Backward Search," Annals of Software Engineering, Special Volume on Requirements Engineering, 3 (1997), pp. 459--475.

12. Maier, T., "FMEA and FTA To Support Safe Design of Embedded Software in Safety-Critical Systems," CSR 12th Annual Workshop on Safety and Reliability of Software Based Systems, Bruges, Belgium, 1995.

13. Mars Microprobe Project homepage, http://nmp.jpl.nasa.gov/ds2/

14. Mars Microprobe, Spacecraft Design, MMP/SPEC-03, D-14222, Rev. A, 11/13/97 and Rev. B Draft, 1/22/98.

15. Pfleeger, Shari Lawrence, Software Engineering: Theory and Practice. Prentice-Hall, 1998.

16. Sommerville, Ian, Software Engineering. Addison Wesley, 1996.